

SW4Lite: Performance Portability using RAJA

Ramesh Pankajakshan
Bjorn Sjogreen



LLNL-PRES-737128

This work was performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under contract DE-AC52-07NA27344. Lawrence Livermore National Security, LLC

 Lawrence Livermore
National Laboratory

SW4Lite

- Proxy for SW4(Seismic Waves 4th order)
- Elastic wave equation using FD
- Explicit 4th order accurate time marching
- Dominant compute kernel is 125 point stencil evaluation(74%)
 - Supergrid 15%
- C++ code derived from Fortran
- Code versions
 - Baseline OMP 3.0
 - CUDA
 - RAJA



RAJA port

- Platform specific memory allocator
 - Malloc
 - CudaMallocManaged
 - Hbwmalloc
- Memory marshalling
- RAJA::forallN for compute kernels

Initial Performance

LOH2 15 Million Grid Points

	CTS-1 Xeon E5 2695 v4	ATS-1 KNL in Quad cache	Coral EA Minsky node
Reference	146	93	17
RAJA	290	220	23
RAJA/Reference	~2	~2.4	~1.4



Triple nested loop kernel(74%)

#pragma omp parallel for

```
for (k..  
    for (j..  
        #pragma ivdep  
        #pragma simd  
        for (i..  
            {  
                STENCIL_EVAL  
            }
```

Triple loop RAJA implementation

```
forallN<EXEC_POLICY, int, int,int>
  (RangeSegment(),RangeSegment(),RangeSegment(),
  [=]RAJA_DEVICE(int k, int j, int i){ STENCIL EVAL });
```

Where EXEC_POLICY is:

```
typedef RAJA::NestedPolicy<
  RAJA::ExecList<RAJA::omp_parallel_for_exec,
  RAJA::omp_parallel_for_exec,
  RAJA::simd_exec >> EXEC_POLICY;
```



Performant Triple loop RAJA implementation

```
forallN<EXEC_POLICY, int, int>
  (RangeSegment(), RangeSegment(),
  [=]RAJA_DEVICE(int k, int j){
    #pragma ivdep
    #pragma simd
    for (int i=istart;i<iend;i++){
      STENCIL EVAL
    }
  });
});
```

Where EXEC_POLICY is:

```
typedef RAJA::NestedPolicy<
  RAJA::ExecList<RAJA::omp_parallel_for_exec,
  RAJA::omp_parallel_for_exec >> EXEC_POLICY;
```



Quad nested loop kernel (15%)

```
for (c...
    #pragma omp parallel for
    for (k..
        for (j..
            #pragma ivdep
            #pragma simd
            for (i..
```

Quad loop RAJA implementation

```
forallN<EXEC_POLICY, int, int,int>
  (RangeSegment(),RangeSegment(),RangeSegment(),
  [=]RAJA_DEVICE(int k, int j, int i){ for(c...STENCIL EVAL }});
```

Where EXEC_POLICY is:

```
typedef RAJA::NestedPolicy<
  RAJA::ExecList<RAJA::omp_parallel_for_exec,
  RAJA::omp_parallel_for_exec,
  RAJA::simd_exec >> EXEC_POLICY;
```

Performant Quad loop RAJA implementation

for (int c...

```
forallN<EXEC_POLICY, int, int>
  (RangeSegment(), RangeSegment(),
  [=]RAJA_DEVICE(int k, int j){
    #pragma ivdep
    #pragma simd
    for (int i=istart;i<iend;i++){
      STENCIL EVAL
    }
  });
}
```

Where EXEC_POLICY is:

```
typedef RAJA::NestedPolicy<
  RAJA::ExecList<RAJA::omp_parallel_for_exec,
  RAJA::omp_parallel_for_exec >> EXEC_POLICY;
```



Post Split Performance

	CTS-1 Xeon E5 2695 v4	ATS-1 KNL in Quad cache	Coral EA Minsky node
Reference	146	93	17
RAJA	144	98	78
RAJA/Reference	~1	~1	~4.6



Quad loop RAJA implementation

```
forallN<EXEC_POLICY, int, int,int,int>
  (RangeSegment(),RangeSegment(),RangeSegment(),RangeSegment(),
  [=]RAJA_DEVICE(int k, int j, int i,int c){ STENCIL EVAL });
```

Where EXEC_POLICY is:

```
typedef RAJA::NestedPolicy<
  RAJA::ExecList<RAJA::omp_parallel_for_exec,
  RAJA::omp_parallel_for_exec,
  RAJA::simd_exec,
  RAJA::seq_exec>,Permute<PERM_LIJK..>> EXEC_POLICY;
```

Inline and Permutation Performance

	CTS-1 Xeon E5 2695 v4	ATS-1 KNL in Quad cache	Coral EA Minsky node
Reference	146	93	17
RAJA	149	96	22
RAJA/Reference	~1	~1	~1.3



Observations

- Prefetch on Coral
- Stride one access
- Thread block configuration
- Vectorization on the CTS-1 and ATS-1 machines
- Cache and flat mode performed the same
- What is acceptable portability ?



Component Runtimes

	BC Comm	Scheme	Supergrid	Forcing	Total
CTS-1	4 (3)	113	23	8	149
ATS-1	23 (25)	49	10	3	95
Coral EA	8 (37)	6	1	5	21



Conclusion

- Loop abstractions viable path to performance portability
- Additional abstractions required to match GPU performance
- Optimizations lagging behind for lambdas
- Abstract memory model

Questions & Comments



Lawrence Livermore National Laboratory
LLNL-PRES-737128



Component Runtimes

	BC Comm	BC Phys	Scheme	Supergrid	Forcing	Total
CTS-1	3.85	1.58	120.33	22.50	7.71	156.01
ATS-1	23.51	2.19	49.17	10.43	3.40	95.44
Coral EA	7.65	0.69	5.68	1.52	4.95	20.77



Final Performance

	CTS-1 Xeon E5 2695 v4	ATS-1 KNL in Quad cache	Coral EA Minsky node
Reference	146	93	17
RAJA	144	98	23
RAJA/Reference	~1	~1	~1.4



Quad loop RAJA implementation

```
forallN<EXEC_POLICY, int, int,int,int>
  (RangeSegment(),RangeSegment(),RangeSegment(),RangeSegment(),
  [=]RAJA_DEVICE(int k, int j, int l,int c){ STENCIL EVAL });
```

Where EXEC_POLICY is:

```
typedef RAJA::NestedPolicy<
  RAJA::ExecList<RAJA::omp_parallel_for_exec,
  RAJA::omp_parallel_for_exec,
  RAJA::simd_exec,
  >> EXEC_POLICY;
```

